Санкт-Петербургский государственный университет

Кафедра компьютерного моделирования и многопроцессорных систем

Калинин Илья Дмитриевич

Выпускная квалификационная работа бакалавра

Визуальная система контроля прохождения спортсменом лыжного спуска

Направление 02.03.02

Фундаментальная информатика и информационные технологии

Научный руководитель:
кандидат техн. наук,
доцент
Гришкин В. М.

Рецензент:
кандидат физ.-мат. наук,
доцент
Погожев С.В

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Fundamental informatics and information technologies
Programing and information technologies

Kalinin Ilya

Visual monitoring system for the passage sportsman ski slope

Graduation qualification work

Scientific supervisor:
Candidate of Engineering Sciences,
Associate Professor Valeriy Grishkin

Reviewer:
Candidate of Physico-Mathematical Sciences,
Associate Professor Sergei Pogozhev

Saint-Petersburg
2017

Оглавление

Введение

Компьютерное зрение является достаточно молодой областью, но динамично развивающейся. Основным предметом изучения в данной области являются методы отслеживания, классификации и обнаружения объектов на изображении. В связи с развитием информационных технологий, увеличения вычислительных мощностей, появлением возможности обрабатывать всё большие объемы данных, компьютерное зрение находит применение во всё большем количестве коммерческих продуктов (проверка качества изготавливаемой продукции, диагностика заболеваний в медицине, распознавание номерных знаков, самоуправляемые автомобили и тд.). В данной работе мы постараемся применить существующие методы компьютерного зрения для решения задачи замера времени прохождения спуска во время проведения горнолыжных соревнований, так как в последнее время широкое распространение получают различные горнолыжные виды спорта, создается огромное количество специализированных горнолыжных трасс для тренировок спортсменов [9] и количество людей, занимающихся данным видом спорта, растет. Популярностью среди спортсменов-любителей пользуется вид горнолыжного спорта под названием “слалом”. Слаломом называется спуск спортсменом на лыжах с горы по заранее определенной трассе длинной 400-500 метров с установленными на ней воротами, ширина которых обычно составляет 3.5-4.5м, а расстояние между ближайшими воротами 0.7 - 15м [12]. Победителем является участник, прошедший установленную трассу за наименьшее количество времени. Соответственно, для определения времени прохождения спуска необходима некоторая система, которая будет начинать отсчет времени в момент пересечения спортсменом стартовой линии и заканчивать в момент пересечения финишной. На данный момент для решения этой задачи используются системы на основе RFID меток, считыватели которых установлены на стартовой и финишной прямых трассы. Данный подход позволяет с высокой точностью засекать время, но оборудование для его реализации является дорогостоящим (3000.00 - 6000.00 USD) и

требует специальной подготовки трассы, что порой может доставлять неудобства, а для спортсменов-любителей и вовсе остается единственный вариант ручного замера прохождения спуска, ибо покупка указанных наборов не всегда вписывается в бюджет. Мы ставим себе задачу создания системы визуального контроля прохождения спуска, для работы которой необходим лишь ноутбук и получаемый с камеры видеопоток. Данное решение позволит нам превратить практически любой лыжный спуск в трассу, готовую для проведения горнолыжных соревнований без огромных затрат на оборудование.

1. Постановка задачи

Как уже было сказано во введении, мы ставим себе задачу создать систему, с помощью которой мы бы могли определять время прохождения спуска отдельного спортсмена по горнолыжной трассе (рис. 1), имея в распоряжении лишь видеокамеру и ноутбук.

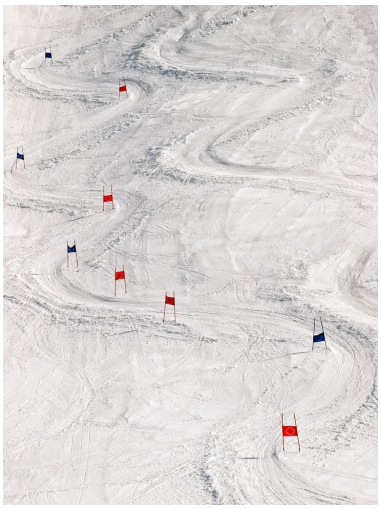

Рис. 1: Пример трассы

Камера при этом будет установлена как показано на рис. 2. На рис. 2

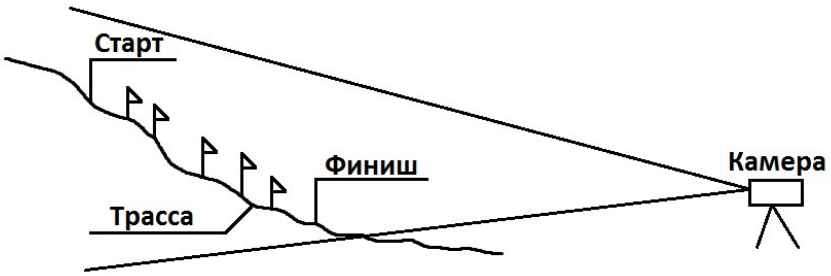

Рис. 2: Вид трассы сбоку, схематичное расположение камеры

видно, что при такой установке камеры, в её объектив попадает целиком вся трасса, вместе со стартовой и финишной линией. При этом положение камеры в пространстве, ракурс и параметры съемки остаются неизменными на протяжении всей работы системы. В нашем распоряжении имеются видеофайлы, на которых запечатлено прохождение спортсменами трассы, при этом видеозаписи содержат запись спуска целиком,

от старта до финиша. При помощи методов компьютерного зрения мы постараемся создать программу, которая позволит нам находить и отслеживать спортсмена на изображении из видеопотока, начинать отсчет времени в момент пересечения им стартовой линии, вести отсчет в момент прохождения трассы и заканчивать в момент пересечения им финишной линии. Необходимым условием будет так же являться то, что в каждый момент времени только один спортсмен должен проходить трассу, а стартовая и финишная прямые должны быть свободны от остальных спортсменов. Поставленную задачу имеет смысл разбить на несколько более мелких подзадач.

1. Создание пользовательского интерфейса, благодаря которому пользователь будет иметь возможность обозначить начало и конец трассы на видеозаписи с помощью мыши, чтобы мы знали, в какие моменты начинать и заканчивать отсчет времени.

2. Предварительная обработка изображения для повышения производительности алгоритма.

3. Локализация области интереса на текущем кадре видеозаписи для последующего распознавания объекта. Данный этап позволит нам выделить все движущиеся объекты на видеозаписи, а последующие шаги позволят выбрать среди них спортсмена.

4. Классификация объекта локализованной области. Этот шаг позволит нам классифицировать найденные движущиеся объекты и отбросить шумы, представляющие из себя снег, дождь, идущих вдоль трассы людей, пробегающих мимо животных и тд.

5. Замер времени. На данном этапе нам необходимо будет определять, пересек ли спортсмен стартовую или финишную линии и останавливать/начинать отсчет в соответствующих случаях.

2. Обзор методов

Для начала рассмотрим существующий метод контроля прохождения спуска, основывающийся на RFID-метках, выявим его преимущества и недостатки, а затем рассмотрим методы, необходимые для решения поставленных нами в п.1 задач.

2.1. Система контроля на основе RFID меток

Радиочастотная идентификация (RFID) - способ автоматической идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся на так называемых RFID-метках. Пассивные метки собирают энергию от радиодиапазонов RFID-считывателя, расположенных поблизости. Активные метки имеют локальный источник питания, такой как аккумулятор, и могут работать на расстоянии тысячи метров от считывателя RFID. В отличие от штрих-кода, метка может не находиться в пределах прямой видимости считывателя, поэтому он может быть встроен в отслеживаемый объект. RFID является одним из методов автоматической идентификации и сбора данных.

RFID-метки используются во многих отраслях промышленности, например, RFID-метка, прикрепленная к автомобилю в процессе производства, может использоваться для отслеживания его продвижения по конвейеру; отмеченные RFID товары можно удобно идентифицировать на складе; внедрение RFID микрочипов в скот и домашних животных позволяет проводить идентификацию животных.

Впервые применение RFID для замера времени в различных видах спорта началось в начале 1990-х годов. RFID помог обеспечить трекинг времени участников в гонках с большим количеством человек, при котором ручной замер данных для каждого участника не представляется возможным.

Во время гонки участники надевают на себя метки, которые впоследствии считывают специализированные антенны, расположенные рядом с дорожкой или на обочинах трассы (рис. 3). Компактные чипы с

коротким радиусом прикрепляют к обуви на липучке или обвязывают креплением вокруг лодыжки. Метки должны находится не далее чем 400 мм от приемника в момент замера, в таком случае удается получить хорошую точность замера времени. Альтернативно, вместо метки может использоваться квадратная 125 мм антенна, надетая на грудь спортсмена не выше 1.25м над уровнем земли.

Рис. 3: 8-канальный приемник рядом с хронометром. У атлета надет чип на ремешке вокруг его лодыжки. Ironman Germany 2007, Франкфурт.

Пассивные и активные RFID-системы используются во внедорожных соревнованиях, таких как спортивное ориентирование, эндуро гонки. Участникам выдаются метки, которые они обычно закрепляются руке, а по окончанию гонки или этапа прикасаются к приемнику, который подключен к компьютеру, и записывают таким образом своё время круга.

RFID также адаптируется многими агентствами по трудоустройству, в которых в качестве квалификационной процедуры применяется тест на физическую нагрузку, особенно в тех случаях, когда количество потенциальных кандидатов может исчисляться миллионами (например, отдел по набору персонала в Индии, сектор полиции и энергетики).

Основное оборудование, необходимое для создания полноценной системы RFID самому:

1. Ридер: считыватель будет сканировать метки, когда они пересекают контрольные точки (425.00-595.00 USD)

2. Антенны: Антенна служит важнейшим элементом, которая будет передавать информацию между метками и считывателем. (425.00 USD)

3. Чипы / метки: метка содержит чип RFID, который хранит и обрабатывает уникальную информацию. Участники гонки будут носить бирку, обычно на нагруднике. (1.00-5.00 USD)

4. Кабели. Кабели подключают антенну к считывателю. (90.00 USD)

5. Программное обеспечение: программное обеспечение будет обрабатывать информацию и отслеживать метки для замера времени участников. (250.00 USD)

6. Система перезаписи информации на метках: необходимо для записи информации на RFID-метку для уникальной идентификации участников гонки. (850.00 USD)

Необходимо учитывать, что антенны и ридеры нам понадобятся в количестве двух штук минимум (для установки на старте и финише), таким образом стоимость, согласно одному из самых популярных поставщиков необходимого нам оборудования RFID Race Timing Systems, составит как минимум 3240.00 USD, а стоимость готового комплекта составит 4,795.00 USD. Поскольку полученная сумма является довольно внушительной, не все спортсмены-любители имеют возможность приобретения подобной системы. К тому же до начала соревнований необходимо проводить установку и настройку всего перечисленного оборудования.
Плюсы метода:

1. Высокая точность замеров времени

2. Существует множество магазинов, где можно приобрести готовые к эксплуатации решения

3. Есть возможность закупить отдельные комплектующие и создать систему самому

Минусы метода:

1. Необходима предварительная установка и перевозка оборудования на места проведения соревнований

2. Высокая стоимость оборудования и программного обеспечения, необходимого для работы метода

2.2. Визуальная система контроля

Для создания предложенной визуальной системы контроля прохождения спуска нам понадобится решить задачу локализации области интереса и задачу классификации найденных объектов. Так как спортсмен на протяжении спуска постоянно находится в движении, было решено свести поставленную задачу локализации к поиску движущихся объектов. Найденные объекты будем классифицировать с целью поиска спортсмена среди движущихся объектов, таким образом отбрасывая шумы. После анализа различных публикаций на тему поиска движущихся объектов в видепотоке [3][2][8] было решено принять к рассмотрению методы вычитания фона, алгоритмы SIFT, SURF, а задачу классификации объектов после анализа работы [7] было принято решать с помощью сверточной нейронной сети.

2.2.1. Методы вычитания фона

Вычитание фона (background subtraction), также известное как определение переднего плана (foreground detection), представляет собой технику, используемую в областях обработки изображений и компьютерного зрения, в которой передний план изображения извлекается для дальнейшей обработки (распознавания объектов и т.д.). Как правило, представляющими интерес областями изображения являются объекты (люди, автомобили, текст и т. д.), которые находятся на переднем плане. После этапа предварительной обработки изображения (который может включать в себя шумоподавление изображения, пост-обработку

и тд.) требуется локализация объекта, для данной задачи и существует данный метод. Вычитание фона - это широко используемый подход для обнаружения движущихся объектов на видео со статических камер. Основой метода является обнаружение движущихся объектов из разницы между текущим кадром и опорным кадром, часто называемым «фоновым изображением» или «фоновой моделью». Вычитание фона в основном выполняется, если рассматриваемое изображение является частью видеопотока. Вычитание фона находит многочисленные применения в компьютерном зрении, например, слежение за автомобильным трафиком. Вычитание фона обычно основано на гипотезе статического фона, которая часто неприменима в реальных условиях. Например, в помещениях отражения или блики солнца могут привести к изменению фона. Аналогичным образом, из-за погодных условий, таких как снег или дождь, применение методов вычитания фона для наружных сцен зачастую не приводит к должным результатам. Именно поэтому в данной работе мы не станем ограничиваться только этим методом, но и применим ряд остальных инструментов для достижения поставленных задач.

2.2.2. Использование разности кадров

Алгоритм обнаружения движения начинается с части сегментации, где передние (движущиеся) объекты сегментируются от фона. Самый простой способ реализовать это - взять изображение в качестве фона и взять кадры, полученные в момент времени t, обозначить их как I(t), сравнить их с фоновым изображением, обозначенным буквой B. Здесь, используя простые арифметические вычисления, мы можем сегментировать объекты просто с помощью техники вычитания изображения для компьютерного зрения, в таком случае для каждого пикселя изображения I(t) взять значение пикселя P[I(t)], и вычесть его с соответствующими пикселями в том же положении на фоновом изображении, обозначенным как P[B]. Математически это можно записать как:

$$P[F(t)] = P[I(t)] - P[B] \tag{1}$$

Предполагается, что фон является кадром в момент времени t. Это разностное изображение будет только показывать некоторую интенсивность для местоположений пикселей, которые изменились в двух кадрах. Хотя мы, казалось бы, убрали фон, этот подход будет работать только в тех случаях, когда все пиксели переднего плана движутся и все фоновые пиксели статичны. Порог (Threshold) ставится на это разностное изображение для улучшения работы метода.

$$|P[F(t)] - P[F(t+1)]| > Threshold \tag{2}$$

Это означает, что интенсивность пикселей разностного изображения «порождена» или фильтруется на основе значения Threshold. [4] Точность этого подхода зависит от скорости движения в сцене. Быстро движущиеся объекты в таком случае могут потребовать более высокие пороги.
Плюсы:

1. Простота реализации
2. Быстрота

Минусы:

1. Необходима очень тонкая настройка пороговых параметров
2. Чувствительность к изменениям освещения

2.2.3. Фильтр среднего (mean filter)

Для вычисления изображения, содержащего только фон, усредняется серия предыдущих изображений. Для вычисления фонового изображения в момент времени t (3),

$$B(x, y, t) = \frac{1}{N} \sum_{i=1}^{N} V(x, y, t - i) \tag{3}$$

Где N - число предыдущих изображений, взятых для усреднения. Это усреднение относится к усреднению соответствующих пикселей в данных изображениях. N будет зависеть от скорости видео (количество

кадров в секунду) и количества движения на видео. После вычисления фона $B(x, y, t)$ мы можем затем вычесть его из изображения $V(x, y, t)$ в момент времени $t = t$ и сравнить его с пороговым значением (Threshold). Таким образом, на переднем плане получим (5).

$$|V(x, y, t) - B(x, y, t)| > \text{Threshold} \tag{4}$$

Где Threshold - пороговое значение. Аналогичным образом мы можем использовать медиану вместо среднего в приведенном выше вычислении $B(x, y, t)$. Необходимо отметить, что использование глобальных и не зависящих от времени пороговых значений может ограничить точность указанных двух подходов. Плюсы:

1. Простота реализации
2. Быстрота

Минусы:

1. Необходима очень тонкая настройка пороговых параметров
2. Лучшая чувствительность к изменениям освещения, чем у алгоритма разности кадров, но недостаточная для решения нашей задачи

2.2.4. Running Gaussian average

В работе [4] Т. Даррелом, А. Пентландом и др. было предложено проводить подгонку вероятностной функции плотности Гаусса (Gaussian probabilistic density function - PDF) по последним n кадрам. Чтобы избежать рассчета PDF с нуля в каждый новый момент времени кадра t, было предложено вычислять среднее значение. PDF каждого пикселя характеризуется средним значением и дисперсией. Ниже приведено возможное начальное условие (предполагая, что изначально каждый пиксель является фоном):

$$\begin{aligned} \mu_0 &= I_0 \\ \sigma_0^2 &= < \text{значение по умолчанию} > \end{aligned} \tag{5}$$

Где I_t - значение интенсивности пикселя в момент времени t. Чтобы инициализировать дисперсию, мы можем, например, использовать дисперсию по x и y из маленького окна вокруг каждого пикселя. Обратите внимание, что фон может меняться со временем (например, из-за изменений освещения или нестатических фоновых объектов). Чтобы учесть это изменение, для каждого кадра t среднее и дисперсия каждого пикселя должны обновляться следующим образом (6):

$$\begin{aligned} \mu_t &= \rho I_t + (1-\rho)\mu_{t-1} \\ d &= |(I_t - \mu_t)| \\ \sigma_t^2 &= d^2\rho + (1-\rho)\sigma_{t-1}^2 \end{aligned} \tag{6}$$

Где ρ определяет размер временного окна, используемого для расчета PDF (обычно $\rho = 0.01$), а d - Евклидово расстояние между полученным средним значением и значением пикселя. Теперь мы можем классифицировать пиксель как фон, если его текущая интенсивность лежит в пределах некоторого доверительного интервала его среднего распределения(8):

$$\begin{aligned} \tfrac{|(I_t-\mu_t)|}{\sigma_t} > k &\longrightarrow \text{Передний план} \\ \frac{|(I_t-\mu_t)|}{\sigma_t} \leq k &\longrightarrow \text{Задний план} \end{aligned} \tag{7}$$

Где параметр k является свободным порогом (обычно $k = 2.5$). Большее значение k учитывает более динамичный фон, в то время как меньшее значение k увеличивает вероятность перехода пикселя с фона на передний план при более мелких изменениях.

В данном варианте метода распределение пикселей обновляется только в том случае, если оно классифицируется как фон. Это сделано для предотвращения застывания новых объектов переднего плана в заднем плане. Формула обновления для среднего изменяется соответственно :

$$\mu_t = M\mu_{t-1} + (1-M)(I_t\rho + (1-\rho)\mu_{t-1}) \tag{8}$$

Где $M = 1$, когда I_t считается передним планом и $M = 0$, когда I_t считается фоном, поэтому, когда $M = 1$, то есть когда пиксель определяется

как передний план, среднее останется тем же. В результате пиксель, ставший пикселем переднего плана, сможет снова стать фоном только когда значение интенсивности приближается к тому, каким оно было перед переходом пикселя в передний план. Однако этот метод имеет несколько проблем: он работает только в том случае, если все пиксели первоначально являются фоновыми пикселями (или пиксели переднего плана аннотированы как таковые). Кроме того, он не может справиться с постепенными изменениями фона: если пиксель классифицируется как передний план в течение слишком длительного периода времени, интенсивность фона в этом месте могла измениться (освещение и т.д.). В результате, как только объект переднего плана уйдет, новая интенсивность фона может больше не распознаваться как таковая.

Плюсы метода:

1. Наибольшая устойчивость к изменениям освещения, по сравнению с остальными алгоритмами
2. Быстрота

Минусы метода:

1. Работает идеально только в том случае, когда все пиксели изначально являются фоновыми пикселями

Из рассмотренных методов наиболее гибким и эффективным являются методы на основе бегущих средних Гауссианов, поэтому далее во время реализации задачи будем рассматривать только их.

2.2.5. SIFT, SURF

SIFT и SURF [8] [6] являются запатентованными алгоритмами поиска дескрипторов (особых точек) на изображении. Они могут использоваться для таких задач, как распознавание объектов, классификация или 3D-реконструкция. Стандартная версия SURF в несколько раз быстрее, чем SIFT и, как утверждают ее авторы[6], более устойчива

к различным преобразованиям изображений, чем SIFT, поэтому далее будем рассматривать только метод SURF. Использование для решения задачи нахождения движущихся объектов данных алгоритмов экстракции уникальных дескрипторов изображения было мотивировано тем, что найдя на двух следующих друг за другом в видеопотоке кадрах уникальные дескрипторы и сравнив их, мы сможем решить поставленную задачу. Проблема возникла при первом же тестировании выдвинутой гипотезы. Оказалось, что ни один из указанных методов не смог обеспечить необходимой производительности (подразумевается, что система обязана обрабатывать в реальном времени видеопоток с разрешением 1280x720 с кадровой частотой 30кадров/сек) на имеющимся оборудовании для тестирования. При установке высоких пороговых параметров для увеличения количества дескрипторов на изображении время обработки каждого кадра превышало допустимое, так как резко возрастало количество объектов, которые необходимо было передавать классификатору. А при снижении значений параметров порогов зачастую инте-

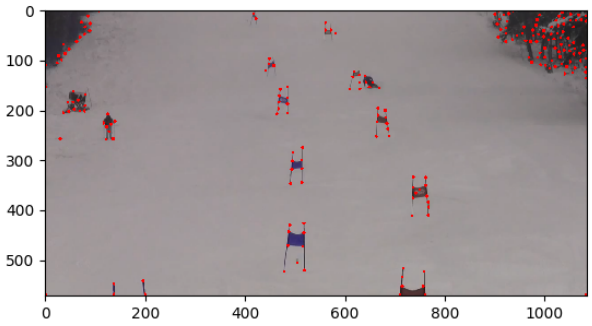

Рис. 4: Пример работы алгоритма SURF (малые пороговые значения, много найденных точек)

ресующий нас объект (спортсмен) и вовсе не имел на себе дескрипторов, таким образом фигура спортсмена довольно часто пропадала из множества движущихся объектов. Указанные алгоритмы подробно описаны в работах [1] [8] [6].

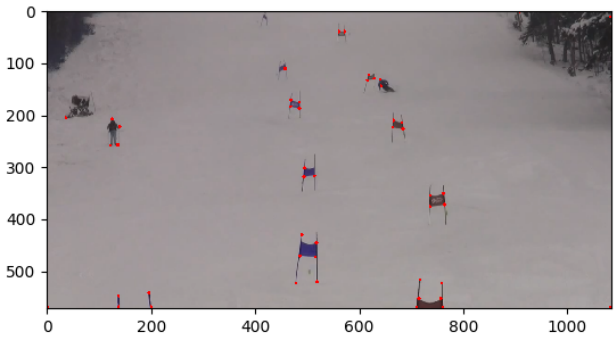

Рис. 5: Пример работы алгоритма SURF (высокие пороговые значения, мало точек)

2.2.6. Сверточные нейронные сети

Сверточные нейронные сети [13] очень похожи на обычные нейронные сети: они состоят из нейронов, которые имеют веса, каждый нейрон получает некоторые входные данные и выполняет над ними скалярное произведение. Вся сеть по-прежнему выражает одну дифференцируемую функцию: от пикселей сырого изображения на одном конце до оценки класса входного изображения на другом. Для сверточных нейронных сетей применима функция потерь (например, SVM / Softmax) на последнем полносвязном слое, таким образом многие методы, которые были разработаны для обучения регулярных нейронных сетей, можно применять и для сверточных нейронных сетей. Использование сверточных нейронных сетей предполагает, что входными данными являются изображения, что позволяет нам закладывать определенные свойства в архитектуру. Нейронные сети получают на вход один вектор и преобразуют его через ряд скрытых слоев. Каждый скрытый слой состоит из набора нейронов, где каждый нейрон полностью связан со всеми нейронами предыдущего слоя и где нейроны в одном слое функционируют совершенно независимо и не имеют общих связей. Последний полносвязный слой называется «выходным слоем», его значения означают принадлежность входного изображения к какому-либо классу. Регулярные нейронные сети не масштабируются до полноразмерных изображений. Например, в CIFAR-10 изображения имеют размер только 32x32x3 (32 широких, 32 высоких, 3 цветовых канала), поэтому один

полностью подключенный нейрон в первом скрытом слое обычной нейронной сети будет иметь вес 32 * 32 * 3 = 3072. Эта сумма кажется управляемой, но ясно, что эта полносвязная структура не масштабируется до более крупных изображений. Например, изображение большего размера, например, 200x200x3, приведет к образованию нейронов, которые имеют 200 * 200 * 3 = 120 000 весов. Более того, мы почти наверняка хотели бы иметь несколько таких нейронов, для быстрого суммирования параметров. Понятно, что такая полная связь является расточительной, и огромное количество параметров быстро приведет к переобучению. Сверточные нейронные сети используют тот факт, что входные данные состоят из изображений, и позволяют выстраивать архитектуру особым образом. В частности, в отличие от обычной нейронной сети, слои сверточной нейронной сети имеют нейроны, расположенные в трех измерениях: ширина, высота, глубина (глубина здесь относится к третьему измерению объема активации, а не к общему количеству уровней в сети). Например, входные изображения в CIFAR-10[1] - это входной объем активаций, а объем имеет размеры 32x32x3 (ширина, высота и глубина соответственно). Как мы скоро увидим, слой нейронов будет связан только с небольшой областью слоя перед ним, а не со всеми нейронами, как было бы в полностью связанном виде. Более того, конечный выходной слой для CIFAR-10 будет иметь размеры 1x1x10, потому что к концу архитектуры сверточной нейронной сети мы сведем полное изображение к единому вектору оценок класса.

Основой любой сверточной нейронной сети является слой свертки. Как уже говорилось ранее, входное изображение - это матрица 32x32x3 со значениями пикселей. Для понимания работы слоя свертки является представление его в виде фонарика. В таком случае свет, излучающий данный фонарик - это область 5x5. Данный "фонарик" движется по изображению, вместе с ним движется указанная область 5x5, называемая рецептивным полем (полем восприятия), а сам "фонарик" в таком случае будет фильтром (нейронном, ядром). То есть наш фильтр — это матрица (такую матрицу ещё называют матрицей весов или матрицей

[1]CIFAR-10 - популярный набор данных, для тестирования на нем различных классификаторов

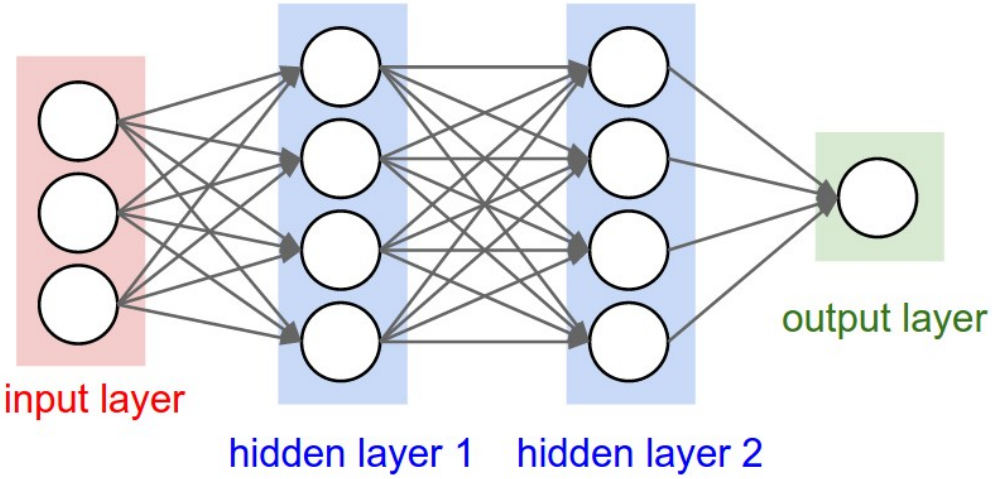

Рис. 6: Пример обычной нейронной сети

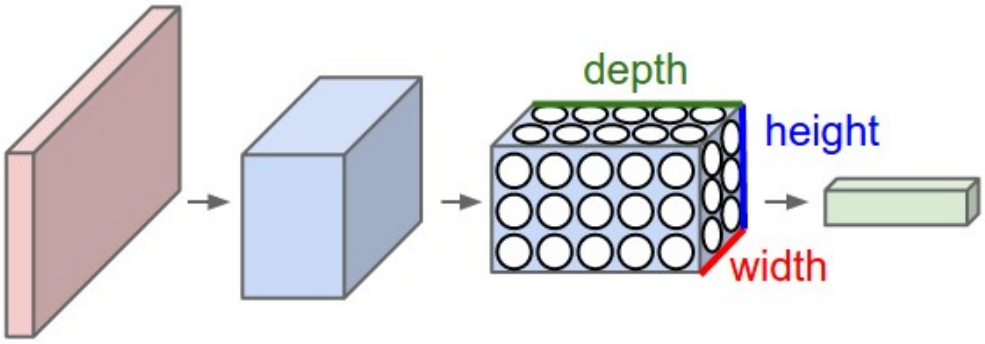

Рис. 7: Пример сверточной нейронной сети

параметров). Глубина у фильтра должна быть такой же, как и глубина вводного изображения (тогда есть гарантия математической верности), и размеры этого фильтра — 5 x 5 x 3. Для примера можно взять позицию, в которой находится фильтр. В нашем случае это будет левый верхний угол. Так как фильтр производит свёртку, то есть передвигается по вводному изображению, он умножает значения фильтра на исходные значения пикселей изображения (поэлементное умножение). Полученные умножения суммируются (всего 75 умножений) и в итоге получается одно число. Оно показывает, что фильтр находится в левом верхнем углу изображения. Теперь повторим этот процесс в каждой позиции, переместим фильтр вправо на единицу, затем еще на единицу и тд. Каждая уникальная позиция введённого изображения производит число. После прохождения фильтра по всем позициям получим

матрицу 28x28x1, которую называют функцией активации или картой признаков. Матрица 28x28 получается потому, что есть 784 различных позиции, которые могут пройти через фильтр 5x5 изображения 32x32. Эти 784 числа преобразуются в матрицу 28 x 28.

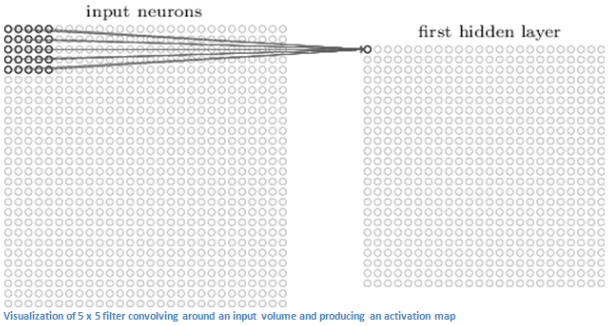

Рис. 8:

Для примера рассмотрим архитектуру самой простой сверточной нейронной сети INPUT - CONV - RELU - POOL - FC:

1. Входной слой [32x32x3] будет хранить необработанные значения пикселей изображения, в данном случае шириной 32, высотой 32 и три цветовых канала R, G, B.

2. CONV (уровень свертки) - его работу мы описали выше. На выходе мы получим матрицу [32x32x12], если мы, например, решили использовать 12 фильтров.

3. Слой RELU (rectified linear unit) - применяет элементную активационную функцию, например, такую как пороговое значение max (0, x). Данная функция показывает хорошие результаты при обучении нейронных сетей и отвечает за отсечение ненужных деталей в канале (при отрицательном выходе).

4. Слой пулинга (POOL) - представляет собой нелинейное уплотнение карты признаков, например, группа пикселей 2x2 уплотнится до одного пикселя, при этом выбирается пиксель, имеющий максимальное значение.

5. Полносвязный слой - вычисляет значения класса, в результате чего мы получаем матрицу размера [1x1x10], где каждое из 10 чисел соответствует значению класса, например, среди 10 категорий CIFAR-10. Как в обычных нейронных сетях, как следует из названия, каждый нейрон в этом слое будет связан со всеми нейронами в предыдущем слое.

2.3. Выводы

Из рассмотренных алгоритмов для задачи нахождения движущихся объектов было решено использовать методы вычитания фона, а конкретно методы на основе running Gaussian average, так они являются наиболее устойчивыми к изменениям освещения и погодным условиям, по сравнению с остальными методами вычитания фона. От использования методов SIFT и SURF пришлось отказаться из-за указанных в пункте 2.2.5 проблем. Для задачи классификации было решено использовать сверточную нейронную сеть.

3. Реализация задачи

Разработка производилась на языке Python, для реализации программного обеспечения были выбраны следующие библиотеки и пакеты: библиотека OpenCV для работы с алгоритмами компьютерного зрения, пакет TensorFlow для работы с нейронными сетями, фреймворк keras для удобной работы со сверточными нейронными сетями. Обучение сети было решено проводить на GPU с использованием пакета Nvidia CUDA, потому что в таком случае мы получим прирост производительности при обучении и многократно уменьшим необходимое для этого время [5]. Все вычисления и тестирование производились на ноутбуке со следующими характеристиками:

CPU	Intel i5-3337U 1.80 GHz
Память	6 ГБ DDR3 1600 МГц
GPU	NVIDIA GeForce GT 740M 2GB

3.1. Описание системы

В процессе разработки было создано программное обеспечение со следующей архитектурой.(рис. 9). Предложенная схема является общим описанием работы программы [11], далее мы рассмотрим каждый этап подробно, а пока выделим основные моменты.

Имеется видеозапись или камера, направленная на лыжный склон таким образом, что в область видимости попадает старт, финиш и сама траектория трассы. Ракурс, как и параметры съемки при этом остается неизменным на протяжении всей работы программы. Первым этапом является получение кадра из видеопотока, данный кадр затем предоставляется пользователю для разметки на нем трассы. Координаты старта и финиша сохраняются в памяти для последующих обращений к ним во время проверки пересечения их спортсменом. Далее происходит этап предварительной обработки изображения, во время которого мы преобразуем изображения из трехнаканльного цветного RGB формата в черно-белое с одним каналом. Затем происходит выделение переднего плана с помощью вычитания фона, нахождение контуров на

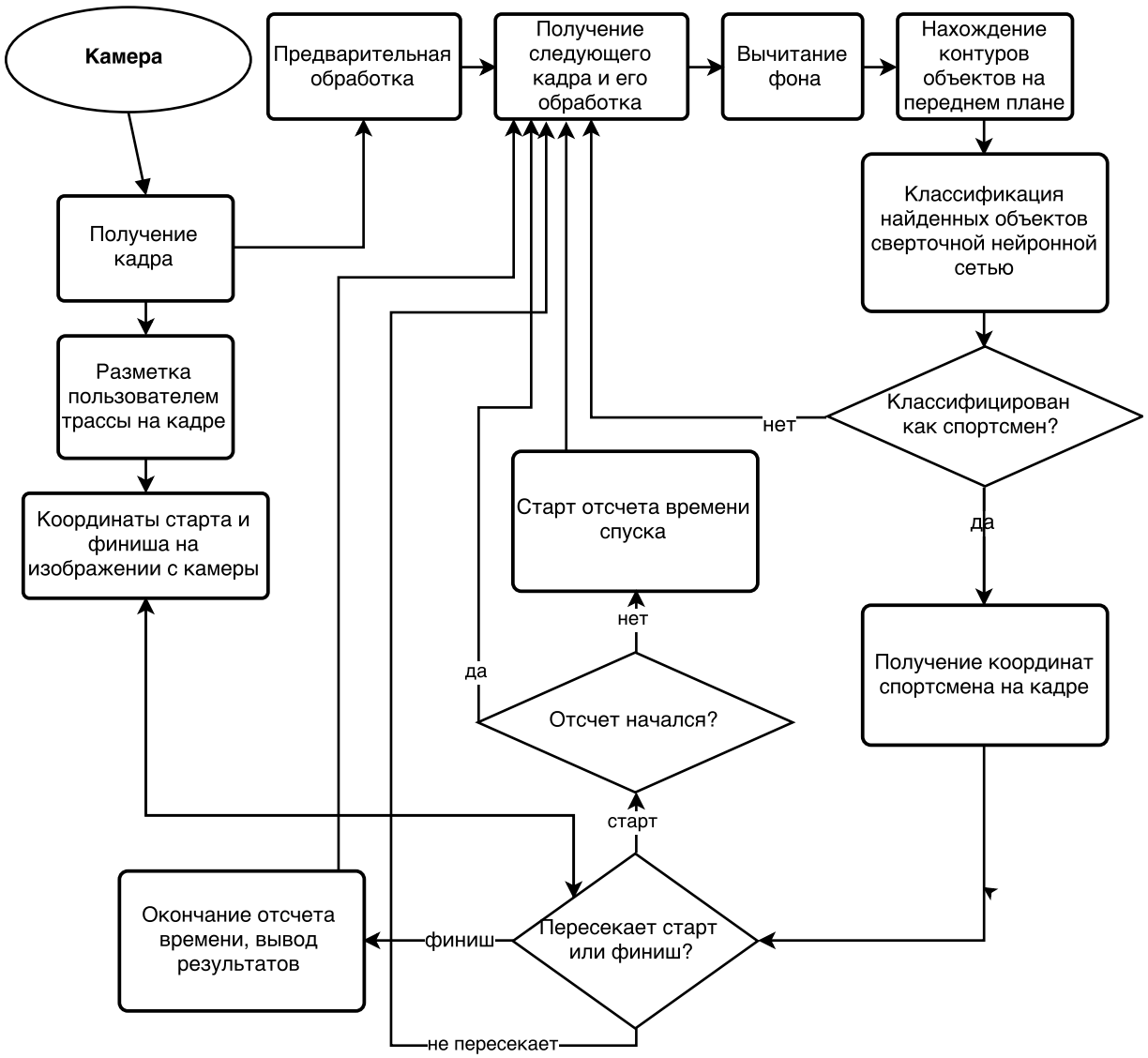

Рис. 9: Общая схема работы разработанного программного обеспечения

изображении с целью поиска интересующего нас объекта (лыжника). На данном этапе в объекты переднего плана попадают все движущиеся объекты на видео, зачастую этими объектами являются капли дождя, снега, качающиеся деревья, флажки на трассе, идущие вдоль трассы люди и тренера. Найденные объекты сохраняются в память и передаются в заранее обученную сверточную нейронную сеть с целью их классификаиции. На данном этапе происходит фильтрация и отбрасывание помех указанных выше. Далее нам необходимо повторять описанные действия для каждого последующего кадра видеопотока и проверять, пересек ли спортсмен стартовую либо финишную черту. В случае пересечения стартовой черты начинать отсчет времени, а в случае пере-

сечения его заканчивать. По заверешению отсчета полученное время выводем пользователю и будем ожидать появления в видеопотоке следующего спортсмена для повторения описанного алгоритма.

3.2. Разметка старта, финиша, предварительная обработка

Как уже было сказано в предыдущей секции, для начала нам необходимо разметить отрезки старта и финиша трассы на изображении. Пользователю предоставляется интерфейс для проведения данной процедуры, с помощью которого он проводит разметку, путем совмещения стартовых и финишных отрезков на экране и на предоставленом ему изображении, которое является первым кадром в загруженном видеопотоке. (рис. 10) В указанном примере верхний отрезок, размеченный

Рис. 10: Пример разметки старта/финиша пользователем

пользователем, является стартом трассы, а нижний концом. Далее следует этап предварительной обработки изображения. В процессе разработки было решено преобразовывать полученные из видеопотока кадры в формате RGB с трехканальным цветом в 8-битовые одноканальные grayscale [2] изображения, содержащие лишь оттенки серого по следующему закону.

$$Y' = 0.299R + 0.587G + 0.114B \tag{9}$$

Применение данного преобразования позволило уменьшить время,

[2]Оттенки серого (градации серого, шкала серого цвета, англ. Grayscale) — цветовой режим изображений, которые отображаются в оттенках серого цвета, размещённые в виде таблицы в качестве эталонов яркости белого цвета. Чаще всего используют ступенчатое изображение равномерного ряда оптических плотностей нейтрально-серых полей.

необходимое для работы метода вычитания фона на 10-50мс для каждого кадра видеопотока.

3.3. Вычитание фона, поиск контуров объектов переднего плана

Полученное изображение с оттенками серого мы будем передовать в обработку методу вычитания фона. В качестве метода вычитания фона было решено использовать алгоритм MOG2, так как в отличии от метода MOG, в котором для каждого пикселя мы используем указанное значение Гауссианов для каждого пикселя, алгоритм MOG2 сам выбирает необходимое количество используемых Гауссианов, следствием чего является повышение устойчивости метода к изменениям освещения, бликам и тд. рассматриваемой сцены, а так же повышение производительности [14]. Результат работы алгоритма приведен на рис. 11. Как мы видим на рис 11, полученное бинарное изображение содержит

Рис. 11: Результат работы MOG2

в себе не только интересующий нас объект, но и огромное количество шумов в виде хлопьев снега, а слева на изображении прослеживается фигура идущего вдоль трассы человека. Данные помехи будут впоследствии отфильтрованы на этапе обработки изображений сверточной нейронной сетью, а пока нам необходимо найти контуры данных объектов. Контуруром является просто граница объекта на изображении. Для распознавания или категоризации объектов используются различные представления контуров (например, цепной код, дескрипторы Фурье, контекст формы). Предполагается, что у нас есть способ сегмен-

тировать объект и найти его границу, что само по себе не является тривиальной проблемой. Исходным источником для поиска контуров обычно является 8-битное одноканальное изображение, в таком случае ненулевые пиксели рассматриваются как 1. Результатом работы алгоритма являются обнаруженные контуры, представленные вектором точек. Контуры будем искать на изображении, полученном в результате работы метода MOG2. Для решения задачи поиска контуров будем использовать функцию бибилиотеки Opencv cv2.findCountours, которая представляет из себя реализацию алгоритма Suzuki85[10]. Найденные данным алгоритмом контуры было решено фильтровать согласно следующим неравенствам (10):

$$\begin{aligned} Length &> (FrameWidth + FrameHeight)/k \\ Area &> (FrameWidth + FrameHeight)/l \end{aligned} \tag{10}$$

Где Lenght - длина периметра контура, вычисленная с помощью функции OpenCV cv2.arcLength(), которая возвращает сумму длин векторов переданного ей контура. Area - площадь, занимаемая контуром, вычисленная с помощью функции cv2.contourArea() - количество точек, находящихся внутри контура, FrameHeight и FrameWidth - высота и ширина кадра, получаемого из видеопотока. k, l - константы, зависящие от разрешения изображений, получаемых из видеопотока, указанные заранее (обычно k=25, l=27 для видеопотока разрешением 1280x720 пикселей). Далее мы будем рассматривать только контуры, которые удовлетворяют неравенству (10) и отбросим контуры, имеющие длину периметра и площадь меньше указанной. Тестирование показало, что данный шаг позволяет нам отфильтровать небольшие шумы в виде маленьких капель дождя и снега, что в будущем позволит уменьшить время, необходимое на обработку каждого кадра видеопотока. Для наглядности покажем найденные контуры на исходном кадре (рис. 12).

Рис. 12: Результат работы алгоритма поиска контуров

3.4. Классификация с использованием сверточной нейронной сети

На данном шаге алгоритма у нас имеется набор контуров объекта переднего плана, т.е все движущиеся объекты. В число этих объектов довольно часто попадают не отфильтрованные на предыдущих шагах помехи (большие крупицы снега; люди, идущие вдоль трассы; качающиеся деревья). Нашей задачей на данном этапе является финальное отделение помех от интересующих нас объектов, поэтому было решено создать классификатор на основе сверточных нейронных сетей. Было принято решение использовать бинарную классификацию, при которой класс 1 - ”позитивные” изображения, на которых присутствует спорстмен, класс 2 - ”отрицательные”, лыжник на изображении отсутствует. На вход нейронной сети будем подавать изображения 80x80 пикселей с трехканальным RGB цветом. Изображения будем получать из наложения полученных в предыдущем шаге контуров на сырой кадр из видеопотока, формировать рамку 80x80 пикселей вокруг центра найденного контура и вырезать полученную область (рис. 13). Для создания

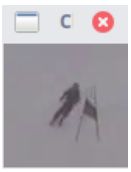

Рис. 13: Пример изображения, подаваемого на вход сверточной нейронной сети

библиотеки образцов нам будет необходимо вручную классифициро-

вать изображения. Было реализовано программное средство, в котором присутствуют все предыдущие описанные шаги алгоритма, на последнем этапе которого мы получаем обрезанное изображение, как показано на рис. 12. Полученные изображения пользователь имеет возможность классифицировать по нажатию клавиши (P - для положительных образцов, F - для отрицательных, space - пропустить и перейти к следующему) сохранить текущее изображение в папку /images/positive, тем самым классифицировав его как "позитивный" образец (рис 14) (содержащий движущегося спортсмена), либо в папку /images/negative, тем самым классифицировав образец как отрицательный (рис 15). Примеры работы программы: Видим движущегося спортсмена, классифициру-

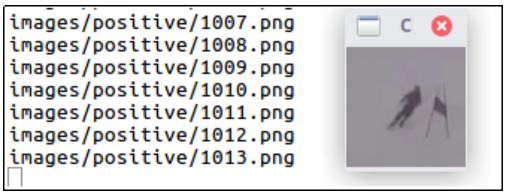

Рис. 14: Пример изображения, которое будет классифицировано как "положительное"

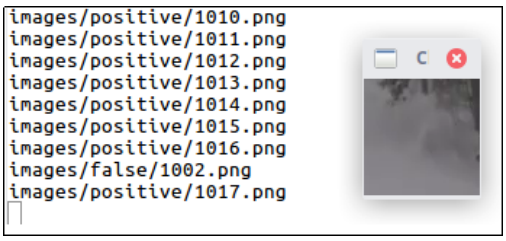

Рис. 15: Пример изображения, которое будет классифицировано как "отрицательное"

ем изображение как положительный образец, нажимаем P, изображение сохраняется в папку положительных образцов. Во всех остальных случаях классифицируем изображение как отрицательный образец, нажимаем F, изображение сохраняется в папку отрицательных образцов. Данная программа разметки образцов позволила нам удобно и за довольно короткое время сформировать бибилиотеку тренировочных и тестовых данных размером в 2000 образцов, 1500 из которых являются

”отрицательными образцами”, а 500 - ”положительными”. Для того, чтобы максимально использовать собранное нами количество данных, мы будем «увеличивать» их с помощью ряда случайных преобразований, чтобы наша модель никогда не увидела дважды одно и то же изображение. Это должно помочь нам предотвратить переобучение и улучшить работу модели. В Keras это можно сделать с помощью класса keras.preprocessing.image.ImageDataGenerator. Будем генерировать новые изображения, используя следующие техники:

1. Поворот на значение в градусах (0-180)
2. Растяжение изображения по вертикали или по горизонтали
3. Перевод цветовых значений. Наши исходные изображения состоят из коэффициентов RGB в 0-255, но такие значения были бы слишком высоки для наших моделей для обработки (с учетом типичной скорости обучения), поэтому вместо этого мы будем использовать значения от 0 до 1, масштабируя исходные значения с помощью умножения на 1/255.
4. Случайное применение сдвиговых преобразований
5. Зумирование в центр изображения
6. Отражение по горизонтали

Рис. 16: Пример сгенерированных изображений

После 12 эпох обучения удалось добиться точности распознавания спортсмена сверточной нейронной сетью на тестовых данных в 93.25%.

Необходимо брать во внимание, что обучение проводилось на данных, полученных из всего двух различных видеозаписей, за неимением

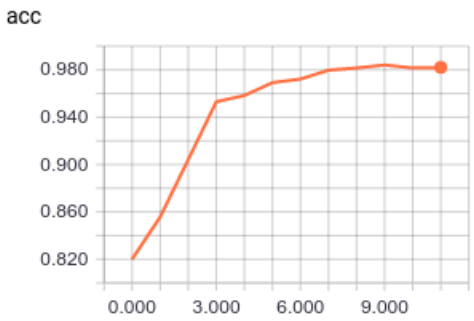

Рис. 17: Точность классификации тренировочных данных (1-12 эпоха)

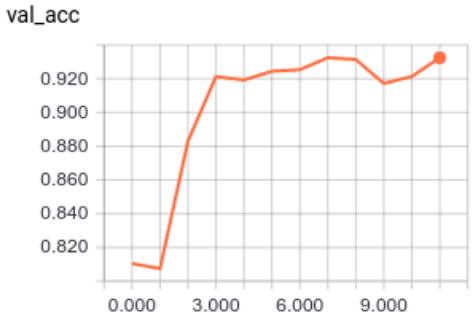

Рис. 18: Точность классификации тестовых данных (1-12 эпоха)

других тестовых данных. Для создания же готовой к установке на горнолыжные трассы системы необходимо будет в будущем провести сбор данных с некоторых других реальных трасс для увеличения количества и разнообразия тренировочных данных.

3.5. Замер времени

Финальным этапом работы алгоритма является определение пересечения найденным спортсменом стартовой линии. Для этого мы последовательно рассматриваем каждый найденный контур в пункте 3.3 на предмет пересечения им стартовой или финишной прямой. Для этого мы ищем пересечения векторов из рассматриваемого в данный момент контура со стартовой или финишной прямой. Начинаем отсчет, если присутствует пересечение со стартовой линией, и заканчиваем, если присутствует пересечение с финишной. После старта таймера будем выводить время(19), прошедшее с пересечения спортсмена стартовой прямой. После пересечения спортсменом финишной линии остановим

Рис. 19: Пример работы программы

таймер и выведем полученное время прохождения спуска на экран. Во время тестирования на двух указанных видеозаписях система правильно провела замер времени прохождения спуска в 21 из 29 случаев.

3.6. Результаты

В ходе выполнения данной выпускной квалификационной работы были получены следующие результаты:

1. Создан интерфейс, позволяющий пользователю размечать прямые старта и финиша.

2. Реализован этап предварительной обработки изображения.

3. Проведен анализ существующих методов нахождения движущихся объектов и реализован один из них.

4. Решена задача классификации объектов с помощью сверточной нейронной сети. Но для создания готовой к эксплуатации системы необходимо будет увеличить размер библиотеки тренировочных образцов, что не удалось сделать в данной работе за неимением большего количества тестовых видеозаписей.

5. Решена задача замера времени прохождения спуска и вывода полученного времени пользователю.

6. Проведено тестирование и показана работоспособность разработанной системы.

Заключение

Разработанное в данной работе программное обеспечение может использоваться для контроля времени прохождения спуска спортсменом на различных горнолыжных трассах. При увеличении библиотеки тренировочных данных с помощью разработанного помощника и дополнительном обучении существующей сети есть возможность увеличить производительность разработанной системы. Даже на слабом процессоре ноутбука разработанный алгоритм позволяет обрабатывать видеопоток с частотой 30 кадров в секунду в реальном времени. В будущем планируется увеличение библиотеки образцов для повышения точности распознавания, а также добавление возможности анализа видеопотоков одновременно с нескольких камер, установленных вдоль трассы. Так же планируется добавить возможность замера времени одновременно для нескольких спортсменов проходящих трассу.

Список литературы

[1] Adesh Hardas Dattatray Bade Vibha wali. Good Features to Track // IEEE Conference on Copmuter Vision and Pattern recognition (CVPR94). — 1994. — Access mode: http://www.ai.mit.edu/courses/6.891/handouts/shi94good.pdf (online; accessed: 12.05.2017).

[2] Adesh Hardas Dattatray Bade Vibha wali. Moving Object Detection and Tracking for Video Survelliance // International Journal of Engineering Research and General Science Volume 2, Issue 4, June-July, ISSN 2091-2730. — 2014. — Access mode: http://ijergs.org/files/documents/MOVING-46.pdf (online; accessed: 12.05.2017).

[3] Adesh Hardas Dattatray Bade Vibha wali. Structural Analysis and Shape Descriptors // International Journal of Computer Applications (0975 – 8887), International Conference on Computer Technology (ICCT 2015). — 2015. — Access mode: http://research.ijcaonline.org/icct2015/number2/icct201538.pdf (online; accessed: 12.05.2017).

[4] Christopher Richard Wren Ali Azarbayejani Trevor Darrell, Pentland Alex Paul. Pfinder: Real-Time Tracking of the Human Body // IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 7. — 1997. — Access mode: https://web.archive.org/web/20070609085400/http://www.cvmt.dk/education/teaching/e06/CVG9/CV/CVG9ExercisePapers/pfinder.pdf (online; accessed: 12.05.2017).

[5] Daniel Strigl Klaus Kofler Stefan Podlipnig. Performance and Scalability of GPU-based Convolutional Neural Networks // Distributed and Parallel Systems Group, Institute of Computer Science. — 2010. — Access mode: http://www.dps.uibk.ac.at/~klaus/Klaus_Kofler_-_Institute_for_Computer_Science_files/GPUCNN.pdf (online; accessed: 12.05.2017).

[6] Herbert Bay Tinne Tuytelaars Luc Van Gool. SURF: Speeded Up Robust Features // European Conference on Computer Vision. — 2006. — Access mode: http://www.vision.ee.ethz.ch/~surf/eccv06.pdf (online; accessed: 12.05.2017).

[7] Karpathy Andrej, Toderici George, Shetty Sanketh et al. Large-scale Video Classification with Convolutional Neural Networks // Conference on Computer Vision and Pattern Recognition. — 2014. — Access mode: http://vision.stanford.edu/pdf/karpathy14.pdf (online; accessed: 12.05.2017).

[8] Lowe David G. Distinctive Image Features from Scale-Invariant Keypoints // International Journal of Computer Vision. — 2004. — Access mode: https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf (online; accessed: 12.05.2017).

[9] NSAA. U.S. Ski Resorts in Operation During 2015/16 Season // NSAA - National Ski Areas Association. — 2016. — Access mode: http://www.nsaa.org/media/275065/Number_of_Ski_Areas_by_Season_1516.pdf (online; accessed: 12.05.2017).

[10] OpenCV. Structural Analysis and Shape Descriptors // OpenCV Documentation. — 2014. — Access mode: http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html (online; accessed: 12.05.2017).

[11] Source code // github. — 2017. — Access mode: https://github.com/kalininilya/skiers-tracker/tree/master/src (online; accessed: 12.05.2017).

[12] Wikipedia. Slalom skiing // Wikipedia, The Free Encyclopedia. — 2017. — Access mode: https://en.wikipedia.org/wiki/Slalom_skiing (online; accessed: 12.05.2017).

[13] Zeiler Matthew D., Fergus Rob. Visualizing and Understanding Convolutional Networks // European Conference on Computer

Vision. — 2014. — Access mode: https://www.cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf (online; accessed: 12.05.2017).

[14] Zoran Zivkovic Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction // Science direct, Faculty of Science, University of Amsterdam, The Netherlands University of Twente. — 2005. — Access mode: http://www.zoranz.net/Publications/zivkovicPRL2006.pdf (online; accessed: 12.05.2017).